DIMENSIONS OF CONTENT ADAPTATION IN THE MOBILE WEB

Eduardo Casais
areppim AG
Bern, Switzerland

1. TRANSCODERS

As the World-Wide-Web Consortium has wound down its activities on the guidelines for content transformation proxies (http://www.w3.org/TR/ct-guidelines), it is instructive to dissect and assess the relevance of the issue underlying this endeavour.

During the second half of 2007, users accessing the Web via their mobile phones were increasingly struck by a severely degraded experience. Formatting of mobile sites was savagely reduced to its simplest expression, and often featured extraneous pre-pended headers or advertisements. Comparable problems affected laptop users connecting to the Internet via wireless cards, with pages containing Javascript throwing odd errors, or images exhibiting unusually poor resolution. Matters became critical for providers of commercial downloadable content (such as games and wallpapers), since customers would no longer be provided with content appropriate for their terminals, and transactions could not be completed because of corrupted protocol parameters. Much of that sorry state of affairs persists to this date.

The culprits were “transcoders” – specialized HTTP proxies (developed by OpenWave, Novarra, GIN and ByteMobile) which intercept requests sent by terminals and responses returned by application servers, and manipulate them in various ways. Transcoders have a reasonable justification: converting desktop-oriented Web pages automatically into a format suitable for mobile devices. The fact that they also aggressively alter mobile-optimized content caused an outcry and led to a couple of initiatives to fix the situation: the “responsible reformatting manifesto” (http://wurfl.sourceforge.net/manifesto) and the W3C guidelines.

Transcoders brought content adaptation into some disrepute. Yet, this process has been fundamental to build services since the inception of the mobile Web. It is therefore useful to review its scope and the role played by components of the content delivery chain.

2. CONTENT ADAPTATION

Content adaptation is the process of producing, selecting or modifying data to satisfy constraints imposed by the service delivery context, which encompasses the following aspects:

- the hardware and software characteristics of the device used to access the service;
- the profile and preferences of the user accessing the service;

- the time at which the service is accessed;
- the place where the service is utilized;
- the performance and status of the network over which the service is delivered.

Services are delivered over a chain of components, which in the WWW typically comprises the following categories:

- the **application server** (which can actually be a complex combination of back-end and front-end servers) provides the service; it is the place where the main business logic and data reside and where the content is usually generated;
- the end-user actually utilizes the service (but does not necessarily initiate it, in the case of push applications) through a **terminal** interacting with the server, possibly generating content in the process;
- a **proxy** serves to establish communications between terminals and servers over the Internet, and forwards requests, responses and associated data via HTTP;
- a **gateway** differs from a proxy in that it interfaces between different protocol domains, most importantly between wireline and wireless networks, or HTTP and WAP.

The following discussion primarily considers conventional browsing – the area where content adaptation and transcoding have been used the longest – but is applicable to AJAX-based Web applications, content download and SOAP-oriented Web services as well.

3. TERMINAL ADAPTATION

Adjustment to characteristics of handheld devices is the obvious focus for content adaptation schemes: the fragmentation of target platforms (browsers, displays, input mechanisms, media capabilities, etc) makes this a necessity. The set of relevant attributes is vast, but can be defined fairly independently from specific applications. Frequently performed adaptations are:

1. Encoding: applying a specific byte representation to a document (e.g. UTF-8 or Shift_JIS for character encoding, base64 for transfer encoding).
2. Conversion: producing a document according to a specific format (e.g. PNG, JPEG or GIF; XHTML mobile profile 1.2, HTML 5.0 or C-HTML).
3. Configuration: constructing a document out of media, markup, CSS, or program fragments, and libraries fine-tuned to the functions of the target run-time environment.
4. Compilation: converting a document into a semantically equivalent form directly interpretable by the device – e.g. turning Java into bytecodes, or WML into WBXML.
5. Clean-up: sanitizing the structure of a document to conform to formal syntactic or structural criteria (e.g. XML well-formedness, safe formatting of special HTML entities).

6. Disinfection: excising from a file suspect executable code (worm, virus, spyware, Trojan) that takes advantage of vulnerabilities in the terminal run-time environment.
7. Image dimensioning: cropping and scaling pictures to fit the target display.
8. Image colour manipulation: palette mapping, reduction to grey or monochrome, dithering, contrast enhancement.
9. URL enhancement: turning a telephone number into a click-to-call link, or a textual address into a map (via the insertion of a static link or a dynamic DOM structure).
10. Linearization: ordering or collapsing presentation elements so that they fit the target display and can be traversed with minimum scrolling.
11. Pagination: dividing a document too large to be stored in the terminal into a succession of smaller interlinked documents.

A server is optimally placed to perform these operations when they concern static terminal properties (stored in a device description repository like WURFL, DeviceAtlas, or DetectRight) – dynamic UAProf notwithstanding – and domain-specific source data (such as proprietary XML dialects, or rare character and multimedia encodings). Conversely, a terminal is in the best position when adaptations depend on dynamic attributes and standard output formats.

Thus, image scaling is best performed on the terminal (which is aware of whether the browser is in full-screen mode or not), but cropping is preferably left to the server (because it has application logic to identify the important parts of a picture). Linearization is more successful on the terminal (since changes between portrait and landscape orientation can be recognized in real-time), but pagination is more appropriate on the server (which can cut overly bulky documents at semantically judicious places). Configuration must usually be done by the server, but for smartphones, a bootstrap routine can detect locally the presence and the exact version of required libraries and style-sheets. Server and terminal erect complementary defences against malware: the former sanitizes user input, cross-checks IP address, `referer`, and cookies, and matches code against virus signatures; the latter enforces the same origin policy, and uncovers viruses through behavioural analyses.

Ultimately though, only server-side business logic determines whether a number corresponds to a telephone or to a fax; it, only, can be endowed with comprehensive algorithmic power to tackle the variety of data formats, of support levels for CSS and DOM, and of known device idiosyncrasies existing in the Internet.

In no case does a proxy or a gateway achieve outcomes superior to those of both terminal and server simultaneously. Even when embodying as much knowledge as the server regarding target devices, it lacks insights about application semantics; while able to deal with as many standard formats as the handheld user agent, it ignores the details of its dynamic properties.

4. USER ADAPTATION

Attribute constituting user profiles or preferences are largely service-dependent, often fuzzier than the technical ones describing a device, and not as well-normalized. Hence, a sample of user-orientated adaptations contains less generic operations:

1. Internationalization: producing content in the natural language and script of the end-user, with associated idiomatical representations of date, time, numbers, currency, etc.
2. Accessibility enabling: adjusting the content to (dis)abilities of the user – for example by rendering text as speech (sound) instead of letters and words (pixels on a screen).
3. Personalization: automatically resorting to stored user-contributed information during standard interactions with services – for instance attaching a vCard to e-mail messages, pre-filling forms with user password and address, playing user-installed ring tones, cross-posting information to various social networking sites.
4. Spam filtering: censoring messages that, from experience about user correspondents and discussion topics, do not seem to constitute legitimate, wished-for communication.
5. Customization: configuring the mode of interaction with the service, such as toggling between an expert or a beginner interface, activating automated interaction sequences (keyboard macros, bookmarklets), or substituting the look and feel (skins).
6. Tailoring: shaping content in function of preferences explicitly stated by the user or inferred from previous interactions with the service. Techniques range from user-defined alerts and RSS polling frequencies, through Netflix or Amazon-like recommendations based on past customer orders, to newsletters individually composed for each recipient.

Server back-ends typically store customer or subscriber information which drives tailoring and internationalization, and enables accessibility and personalization. But even in the case of Internet cloud applications, user agents rely upon a wealth of parameters managed locally – many of which leave the mobile phone only during backups. Usability and accessibility configurations ultimately take place on the terminal. Apart from spam filtering, proxies and gateways are at a disadvantage since they do not actively manage user information. Their lack of semantic context about the services under use entails less precise adaptations, for instance when tailoring advertisements based on browsing patterns.

5. TIME ADAPTATION

Applications built upon temporal data are natural candidates for adaptations along the time dimension, but even when they are not, the association of service interaction events with timestamps also leads to interesting transformations:

1. Filtering: temporal data is selected to show just the relevant items for the time at which the service is accessed. A calendar displays entries from the current day or week onwards; a timetable, departures or programmes valid in the near future; a ticker, the most recent sports results or stock exchange figures.
2. Date or time pre-setting: filling in user input with appropriate default values – e.g. assuming the date to issue an electronic funds transfer to be the next day.
3. Time-of-day modes: altering the presentation of information in synchrony with the passage of time. The automatic switch between “day mode” and “night mode” in car navigation systems is well-established; the phone model “Satio” from SonyEricsson includes a similar option that flips wallpapers depending on the time of day.
4. Reordering: presenting service entry points (such as links or menu items) sorted according to their presumed temporal relevance. Thus, a Web portal might position bus and train timetables prominently on a page of useful links before rush hours to and from work, and do likewise for TV and cinema programmes during the evenings.

Meaningful time adaptations require a semantic context tied to server-side business logic, or to full-fledged client applications; here, intermediate network elements play barely a role.

6. PLACE ADAPTATION

Location-based services have long been a prime contender to become the “killer application” in the mobile Internet. There are various ways the position or movement of the terminal may serve to shape the information presented to the end-user:

1. Profile-based setup: switching sets of properties through the activation of profiles abstracting out the user location. Thus, phone user interfaces (e.g. sound levels), and unified messaging systems (e.g. delivery of communication as voice, SMS, e-mail) adjust to the current user status – “in a meeting”, “outdoors”, “at home”…
2. Augmented communication: attaching detailed location status to messages sent by the terminal. The booking application TaxiMagic sends terminal GPS coordinates as pick-up location to cab companies, which feed them into their fleet dispatching system.
3. Mapping: a representation of the location, surroundings, and movement of entities of interest constitutes the information displayed to end-users. In a navigation system, the entity of interest is the device itself. In a geosocial network such as Loopt, entities of interest comprise designated “friends” who keep track of their respective whereabouts.
4. Localization: selecting exactly those back-end data that are relevant for the current location of the end-user. E-commerce sites let customers choose a country from a list and then restrict the Web product catalogue to those

items marketed in that country. Tools like the ip2country database (http://ip-to-country.webhosting.info) streamline the procedure by deducing the location directly from the IP address of the terminal. Specialized directories return links (to stores, hotels, restaurants, bus stops…) in the vicinity of an address provided via GPS or entered manually by the end-user.

5. Augmented reality: enhancing the information gathered in situ by the terminal with location-relevant data retrieved from servers. Wikitude and Layar, for instance, overlay video images with details, queried from a database, about the object being observed – an approach that has given rise to innovative forms of interactive travel guides.

6. Gesture input: controlling applications by moving and rotating the terminal. The Wii play station popularized this class of user interfaces; manufacturers of mobile phones followed by integrating motion sensors and associated libraries into their products.

In most cases, terminal and server must cooperate to bring about location-specific transformations; in fact, complex ones are impractical without advanced client software. Beyond determining the position of terminals (for instance by interfacing with a GMLC), further network elements have little to offer in terms of application-relevant place adaptations.

7. NETWORK ADAPTATION

Network performance strongly affects the usability of mobile Web applications. Several techniques help deliver content efficiently in wireless environments – which exhibit marked differences regarding capacity, congestion and delays compared to the wireline infrastructure.

1. Compaction: converting a document to a syntactically equivalent, but smaller representation. Specialized utilities such as minify and Smushit variously eliminate superfluous spaces and comments in code, CSS and markup, delete optional ending tags in HTML, rewrite identifiers to shorter strings, or optimize images.

2. Compression: transforming a document into a denser binary representation that can be reversed to restore the original format. GZIP is the standard lossless compression scheme over HTTP; its usage is recommended to reduce the payload of transactions.

3. URL shortening: rewriting links to reduce the length of references in pages and in HTTP requests. This popular service, provided by the likes of mtny.mobi, mobiletinyurl, or Delivr, has the additional advantage of circumventing bookmark cut-offs in browsers.

4. Image degradation: reducing the quality of a picture (size, resolution, bit depth) to limit the payload over a connection with narrow bandwidth. While one expects degradation to proceed gracefully, some implementations, oblivious of actual network conditions, have applied this operation systematically (for instance at Vodafone Germany).

5. Variant selection: choosing the version of a file that respects constraints on network performance, assuming decreased media quality. Thus, users can be presented with the alternative of downloading a full-screen HDTV feature or a reduced QCIF video clip, a document with elaborate layout or its text-only ASCII equivalent.
6. Flow control: smoothing the delivery of a data stream across variations of network throughput, latency and reliability. Audio and video streams form the major application domain, but a dynamic Web application might also throttle the frequency of AJAX updates from a server to cope with network congestion.

These operations do not require much semantic context about the content being adapted, and can therefore be undertaken by any element with sufficient processing power. A gateway interfaces closely with units of the core telecommunication network (access router, GGSN, SMSC), and can thus measure transmission parameters on the wireless and on the wireline leg of the end-to-end Internet connection – potentially for each individual client. It is therefore ideally placed to resolve performance discrepancies between these two environments.

8. OTHER TRANSFORMATIONS

Further content transformations in the mobile Web can hardly be construed as adaptations:

1. Obfuscation: converting a document to an equivalent, but lexically illegible form. The usual goal is to hinder the reverse-engineering of delivered content.
2. Censorship: excising portions of the communication flow between a client and a server. This can affect requests (i.e. preventing visits to specific Web sites) as well as responses (i.e. expurgating content deemed inappropriate).

Transcoders are known to manipulate content in highly contentious ways. Various combinations of the following operations have been implemented in commercial deployments:

- elimination of corporate-specific styling from Web pages;
- insertion of operator-branded banners and navigation bars;
- deletion of advertisements which firms rely upon to monetize accesses to their sites;
- insertion of advertisements resulting in revenue for the telecom operator;
- rewriting all URL, so that HTTPS links get forcibly re-routed through the operator transcoder. As a consequence, there is no longer any end-to-end TLS/SSL connection between the client and the server. Rather, there is a first leg between the terminal and the transcoder, and a second one between the transcoder and the server. Information appears in clear on the transcoder, so privacy and integrity are no longer guaranteed.

Such mutations do not bridge a gap between the capabilities required by services and those afforded by terminals. They do not compensate any deficiency in the delivery context – in fact, they introduce defects regarding look-and-feel, usability and

security into applications. In economics, the technical term describing this approach is “value extraction” (as opposed to “value creation”); it is no wonder that these features generated a hefty, enduring controversy.

9. AN EXAMPLE OF TRANSCODING

The core argument in favour of transcoders is that they enable owners of handhelds to browse the vast realm of desktop-oriented, mobile-incompatible Web sites. Concretely, what does it take to achieve at least a 90% success rate in converting Web content into a form not just simply displayable, but truly usable on mobile devices?

A look at e-comics provides an answer. Japanese developers have devoted a surprising amount of technical ingenuity to elaborate schemes for reusing existing Web mangas on mobile phones. A recent paper (Kohei Arai, Herman Tolle, “Automatic e-comic content adaptation”, International Journal of Ubiquitous Computing, 1 (1), May 2010) describes the necessary steps:

1. Retrieve the Web page, parse it and extract the comics image within it.
2. Insert white space to separate potential frames that seem to overlap.
3. Identify the individual frames in the comics image through thresholding, colour inversion and connected component labelling algorithms.
4. Apply similar techniques to detect the text balloons inside each frame.
5. Apply a text recognition algorithm to balloons and translate the outcome if necessary.
6. Cut the original image into separate pictures along detected frames.
7. Scale each frame and reduce its colour adequately for the target terminal.
8. Process any relevant textual components of the original page.
9. Include the translated balloon texts.
10. Convert each frame into a suitable image format.
11. Order and encapsulate all picture and text fragments into a final markup document.

Evidently, producing appealing content requires a significant amount of domain-specific knowledge within the service platform – such as image processing routines finely tuned to the structural peculiarities of e-comics. Crucially, the method described above is successful because it exclusively processes e-comics pages. A proxy traversed by content flows from widely different sites would have to incorporate substantially more complex logic to discriminate amongst pages containing comics, city maps, photographs, organigrams, or charts, and activate the specialized e-comics adaptation engine only when appropriate. In conclusion, approximating 100% correct content generation implies that all adaptations are embedded in the system delivering the original service, not retrofitted onto it.

10. OVERALL ASSESSMENT

The table below assesses the suitability of each element in the service delivery chain with respect to the five kinds of content adaptation, from 1 (best) to 4 (worst). Ties are indicated by half units. The figures in each category were arrived at by cumulating ordinal evaluations for all individual operations listed in the previous sections into ranks, not continuous scores. The intent is to assess entries at a high-level, and avoid spurious comparisons based on an illusory numerical accuracy – or on figures that could be incommensurable across various categories.

This assessment reflects the technical situation in late 2010; a decade earlier, terminals played a marginal role in content adaptation. Thanks to ever more powerful client software, the importance of end-user devices is bound to expand, and will perhaps become predominant.

Adaptation	Element of the delivery chain			
	Server	Proxy	Gateway	Terminal
Device	1	3,5	3,5	2
User	2	3,5	3,5	1
Time	1	3,5	3,5	2
Place	2	4	3	1
Network	2	3	1	4

Clearly, content adaptation is essentially a matter for terminals and servers – with one striking exception: network adaptations carried out at gateways have the potential to improve quality of service. This fits the mission of telecom operators well, whose core competence lies in the optimization of the load of the "bit pipe". For example, algorithms can compute on a gateway the (dynamically evolving) bit-rate perceived by a terminal, and adjust image properties so that the download time remains constant under variable picture quality; this technology was developed and demonstrated a decade ago. It is all the more surprising to observe how crudely image degradation is currently put into effect at supposedly experienced carriers.

The need for mobile phones to access Web sites designed exclusively for desktop computers is genuine, but occasional, as ever more sites get endowed with facilities to generate content for multiple targets. So are the requirements to provide a degree of backward compatibility (allowing older devices to access sites designed for newer models) or of forward compatibility (enabling modern devices to access legacy content, e.g. an iPhone to read a WML deck). Clean-up and disinfection capabilities are pertinent – after all, every ISP includes virus and spam filters in its e-mail offering – but their value probably accrues to firms like McAfee or F-secure, not transcoder vendors. In short, service-agnostic transcoding proxies carrying out generic content adaptations fulfil an increasingly subsidiary, though worthwhile function.

11. HYBRID APPROACHES

Servers, proxies, gateways and terminals have complementary strengths; integrating these elements should therefore help mitigate their respective shortcomings during

content adaptation. Successful configurations are now well-entrenched in the mobile Web:

1. Proxy + terminal.

   Distributed browsers (UCWEB/UC Browser, Skyfire, Opera Mini) have rapidly risen to prominence because they bring two essential benefits: low-end devices can visit Web sites too complex for their native browsers; and all types of handhelds get improved performance through massively reduced transfer payloads over the air.

   In this configuration, the proxy renders Web pages into a proprietary, highly compressed format sent to a special browser (usually written in Java) installed on the terminal. The proxy also takes over some tasks that cannot be executed on the device – such as running certain kinds of Javascripts, refreshing pages, or managing cookies.

   Because client software and proxy are co-designed and are exactly aware of each other's capabilities, they can adapt content to a quality much higher than possible by a general-purpose transcoder. Thus, there is no uncertainty as to what happens with the title of a page, whether the number of form fields is restricted, how tables are laid out, what exact CSS features are understood, or how long bookmarks can be.

   Proxy-based browsers are not universally capable though: support for Javascript functions, fonts, plugins, access to local storage or to location-based information may be lacking. In addition, just like transcoders, those browsers break end-to-end secure connections normally established through HTTPS URL over TLS/SSL, and allow similar misuses as those listed in section 8 (thus, version 2.5 of the Bolt proxy-based browser forcibly inserts its own advertisements into third-party Web pages).

2. Server + proxy.

   The generic facilities of content adaptation proxies can be manually customized to recognize the page organization of specific Web sites. This approach, embodied in services like mobifyme, goMobi, or mobile plugin packages for blogging platforms, constitutes an efficient way to mobilize existing desktop-oriented services without developing a mobile application from scratch.

   Here, an application designer specifies (often with a semi-WYSIWYG tool) which elements of a page are to be retained and which can be eliminated, in which order they must appear, and corrects the final presentation by attaching a few extra CSS directives. The customizations are recorded by the proxy (for instance as XSLT processing instructions) and replayed during transcoding.

   Since it is parameterized with an accurate model about how information is arranged in the source site, the transcoder no longer relies upon abstract linearization heuristics that often return illogically structured mobile versions of Web pages. A drawback of the method is that it remains tied to the

control flow of the original desktop-oriented site, which may entail deeply nested page hierarchies. Furthermore, cropping and rescaling does not always make desktop media directly usable on mobile phones – maps ought to be zoomed in or out, charts redrawn, comics split, etc.

3. Server + terminal.

   Whether it is with Web pages taking advantage of Javascript and AJAX, or with compiled programs downloaded onto the terminal, not-so-thin-client/server approaches to mobile services are widespread. Applications become potentially fully aware of terminal capabilities (including dynamic capabilities like display orientation or available memory), of user profiles (including phone book), of device location, and of perceived network performance at both ends of the connection.

   From the viewpoint of content adaptation, this combination currently offers the most benefits – but it forces developers to confront thorny issues: installation and versioning of client software, increased difficulties to ensure portability across terminal run-time environments, growing security risks through contamination by spyware. Nonetheless, current best practices for analysing the service delivery context leverage both a server-side infrastructure (usually built around a device description repository) and client-side mechanisms (typically Javascript code or CSS media queries).

   The overview of hybrid systems reinforces previous conclusions: apart from gateways optimizing network load, the space for pure players in the area of transcoding proxies is rather narrow. Hence, it is understandable that dotMobi turned its InstantMobilizer transcoder into a tool for application-driven content adaptations in the form server+proxy above, or that Nokia recast transcoding technologies acquired from Novarra as the distributed Ovi browser.

12. W3C CONTENT TRANSFORMATION GUIDELINES

After three years of effort, the W3C Content Transformation Guidelines will remain an informational Working Group Note. The failure to collect implementation reports (only dotMobi filed an Implementation Conformance Statement (ICS) on its service “Instant Mobilizer”) precluded promoting the guidelines to a normative W3C Recommendation. Neither could a test suite be set up to assess the conformance of transcoder deployments. This leaves the “Manifesto for responsible reformatting” as the only prescriptive document with significant formal endorsements from the mobile industry. However, these originate from application and platform vendors, not telecom operators – a gap that the W3C guidelines were meant to close.

This disappointing outcome cannot be due to some inability of telecom operators; after all, they know very well the features and configuration of their deployed transcoders, had largely enough time to prepare themselves for the final version of the document, could draw some marketing advantage from being compliant with the guidelines, and filling ICS is a regular task in telecoms. As for test suites, they must

exist at the transcoder vendors, since otherwise how would they even check that their systems implement what they are supposed to do?

Presumably, no ICS were forthcoming because the main parties lost interest, and turned their attention to new fashionable mobile technologies – first widgets, then application stores for high-end devices. After market consolidation, when Nokia took over Novarra, support for a wide spectrum of mobile phones and content adaptation ceased to be viewed as burning issues.

This attitude is mistaken. The current breed of transcoders emerged in 2007, in the wake of the iPhone introduction and the perceived need to make the "full Web" also accessible to feature phones running WAP 2 (based on XHTML mobile profile, PNG, and WCSS). However, it was preceded around year 2000 by a generation of content adaptation engines converting Web pages to Unwired Planet/Phone.com/Openwave (HDML, BMP), i-Mode (C-HTML, GIF), and WAP 1 (WML, WBMP) formats. The lessons learned while operating those earlier transcoders were evidently lost or ignored later on. If history is any guide, we shall experience yet another avatar of content transformation in approximately three years. Perhaps advanced electronic readers will have become so ubiquitous by 2013 that there will be an urge to convert rich e-book formats to their mobile Web equivalents (relying upon HTML 5.0, Javascript, CSS and SVG). Meanwhile, even smartphones connected via broadband wireless networks will not enjoy immunity from the effects of transcoders – such as the traffic optimization measures applied by Verizon in its 3G network (http://support.vzw.com/terms/network_optimization.html). In any case, the same difficulties tackled previously will have to be dealt with again. Neglecting knowledge about content adaptation assembled in the now respectably long history of the mobile Internet ensures that the process will prove as hurtful to the concerned parties as it has been in the recent past.

13. CONCLUDING REMARKS

The investigation of solutions to implement content adaptation would require a whole series of lengthy reports. A suggestion is to use the dimensions of content adaptation surveyed in this paper as basis for requirements on the Web service platform. Here are some further pointers:

- A tutorial explaining how to detect device features, with references to CSS media queries, device description repositories, user agent switching algorithms, Javascript queries and general strategies for terminal adaptation can be found at http://wiki.forum.nokia.com/index.php/Device_and_feature_detection_on_the_mobile_web.

  Another tutorial examines mechanisms to retrieve and use device location at http://mobiforge.com/developing/blog/location-location-location.

- The CMS storing source data from which Web pages are generated for multiple platforms is a core component of the service platform. Developers must pay attention to its capabilities regarding data types, multi-lingual support, character encodings, collating sequences, markup formats, and

audio and video media profiles. One can start with an evaluation at http://www.cmsmatrix.org.

- Mobile Web service platforms (WAPPLE – http://wapple.net, WALL/WNG – http://wurfl.sourceforge.net, Volantis – http://www.volantis.com, Netbiscuits – http://www.netbiscuits.com, MyMobileWeb – http://morfeo-project.org, mobileelements – http://www.mobileelements.com, among others), must cater for terminal adaptation and are therefore built upon a device description repository (notably WURFL – http://wurfl.sourceforge.net, DeviceAtlas – http://deviceatlas.com, DetectRight – http://www.detectright.com); perusing the data dictionary of the DDR gives a good idea of the amount of details that can go into terminal adaptation.
- There is a bewildering menagerie of content adaptation tools – from generic XSLT processors and the Swiss army knife of image manipulation ImageMagick (http://www.imagemagick.org), through utilities such as tidy (http://tidy.sourceforge.net) for cleaning up markup and converting between HTML and XHTML, or web2wap for supplying legacy phones with Web content, to authoring software with accessibility remediation facilities (http://www.w3.org/WAI/ER/tools/complete), service providers' API for querying and adding location or mapping information to applications, graphics packages, and more. Many of these components can be plugged-in into or are an integral part of mobile Web application frameworks.

Finally, even when operating a full-fledged framework capable of multi-target content generation, it is valuable to follow best practices that embody accumulated experience on how to ensure portability, usability and performance in mobile Web applications. The most accessible are the GAP (http://www.passani.it/gap), the dotMobi developers' guide (http://mobiforge.com/directory/resources-communities/books-magazines/dotmobi-mobile-web-developers-guide), and W3C recommendations (http://www.w3.org/TR/mobile-bp); hardware vendors and telecom operators also publish their own, useful guidelines, with special consideration for the distinctive characteristics of their products or service offerings.

REFERENCE

Eduardo Casais: *Dimensions of content adaptation in the mobile Web*, technical paper, areppim AG, Bern, Switzerland, 2010-10-05, rev. 2011-02-21, 14 pages.

ABOUT THE AUTHOR

Eduardo Casais has been working on mobile Web technologies since 1997. He led the development of content adaptation facilities in the Nokia WAP Gateway. He was also involved in projects dealing with transcoders for WWW on TV set-top-boxes. Eduardo Casais has been an invited expert to the Mobile Web Best Practices Working Group of the World-Wide-Web Consortium, where he participated in the elaboration of the Content Transformation Guidelines.

ABOUT AREPPIM AG

areppim AG develops Internet applications, with an emphasis on the display of quantitative information. The site http://www.areppim.com publishes data on a wide range of topics, presented as intuitive, content-rich charts and often accompanied by concise analyses. Naturally, data can also be accessed with mobile phones at http://mobile.areppim.com through a no-frills, mobile-optimized interface.

ADDRESS

areppim AG
Wankdorffeldstrasse 102
P.O. Box 261
CH-3000 Bern 22
Switzerland

e-mail: info@areppim.com